**Problem Statement**

***The task defined below will assess the following abilities:***

* ***IM/IT systems development***
* ***Effective written communication***
* ***Teamwork***

**Task:**

Transport Canada (TC) is looking for an API solution that will load a JSON file extract of data contained in the Vehicle Recall Database (VRD) API. Using this data, each team member will need iterate through each record value set to call the VRD and extract additional data elements described further below. The final results should be stored and used to provide the input for the next API, along with using it for the get functionality.

Each team member must create a new API with:

1. *A POST* request that can receive JSON input
2. A GET request to retrieve ALL data
3. A *GET* request to retrieve a single record if provided with a record identifier

All team members must also work together as a team to create a single cohesive web site to:

1. Load a JSON input files
2. Read and Post loaded JSON data to the appropriate API.
3. Create a simple search function to demonstrate a GET of the data once it has been posted and processed by the API.

TC will provide the list of mandatory fields and any associated data at the start of this exercise. This initial data should be used as input for API 1 (POST) via the web site. From that point forward, data retrieved from each API will used as input into the next API. For example:

* GET API 1 results and POST to API 2
* GET API 2 results and POST to API 3
* GET API 3 results and POST to API 4
* GET API 4 results and POST to API 5

All API calls to *POST* must handle missing and/or null values to account for any instances where the team cannot successfully complete all API’s. For example, API 2 results should feed API 4 if API 3 was not successfully completed.

(VRD).   <https://open.canada.ca/data/en/dataset/1ec92326-47ef-4110-b7ca-959fab03f96d>

[

  {

    "RecallNumber": “2016253”,

    "ManufactureName": "FORD",

    "ModelName": "FUSION”

  },

{

    "RecallNumber": “2016527”,

    "ManufactureName": "FORD",

    "ModelName": "MUSTANG”

  }

]

* API 1: Using the JSON file supplied by TC, create a *POST* function to accept the JSON input. For each Recall Number, make a call to the VRD API and extract the MANUFACTURER\_RECALL\_NO\_TXT.
  1. Create a new JSON file that adds the MANUFACTURER\_RECALL\_NO\_TXT to the JSON file.
  2. Create a *GET* function that can load the resulting JSON file, allowing for Retrieval and Display of all data, and a “Search by Manufacturer Recall Number”.
* API 2: Using the JSON file output from API 1, create a *POST* to accept the JSON as input. For each Recall Number, call to the VRD API and extract the CATEGORY\_ETXT and/or CATEGORY\_FTXT.
  1. Create a new JSON File that appends the CATEGORY\_ETXT / CATEGORY\_FTXT to the input JSON.
  2. Create a *GET* function to load the resulting JSON file, allowing for Retrieval and Display of all data, and a “Search by Category”.
* API 3: Using the JSON file output from API 2, create a *POST* to accept the JSON as input. For each Recall Number, make a call to the VRD API and extract the SYSTEM\_TYPE\_ETXT and\or SYSTEM\_TYPE\_FTXT.
  1. Create a new JSON file that appends the SYSTEM\_TYPE\_ETXT / SYSTEM\_TYPE\_FTXT to the input JSON.
  2. Create a *GET* function to load the resulting JSON file allowing for Retrieval and Display of all data, and a “Search by SYSTEM TYPE”.
* API 4: Using the JSON file output from API 3, create a *POST* to accept the JSON as input. For each Recall Number, make a call to the VRD API and extract the NOTIFICATION\_TYPE\_ETXT and/or NOTIFICATION \_TYPE\_FTXT.
  1. Create a new JSON File that appends the NOTIFICATION \_TYPE\_ETXT / NOTIFICATION \_TYPE\_FTXT to the input JSON.
  2. Create a *GET* function to load the resulting JSON file allowing for Retrieval and Display of all data, and a “Search by Notification Type”.

The APIs can be built using any technology, language, and framework(s) preferred. Please note, however, that your choice of technologies must be easily accessible for verification and testing by the Transport Canada Evaluation Team. Teams are discouraged from using custom languages or off-the-shelf toolkits that require purchase or special licensing. Each team member must code their API individually but communication with other team members is encouraged to maintain consistent API inputs and outputs.

Each team must work together to create a single website that demonstrates the APIs built and controls the order in which the APIs run. The team should choose how to construct this website and the web framework(s) used for the demonstration.

Your code should be delivered in either ZIP format or hosted in GitHub. Your team’s submission must include a ReadMe file with the following information:

1. Installation process for each API And the team’s website,
2. If the website is hosted and available to the evaluation team, please provide a working URL in the ReadMe file; otherwise, the evaluation team will build and run your website and API’s based on your detailed instructions.
3. Include all detail for any prerequisite software and versions for the evaluation team
4. Provide all build steps needed to build your team’s website and API’s from a fresh clone, including details about the tools needed to build the code and configuration steps to successfully produce a clean build. If necessary, complex build instructions can be documented in a separate file that can be included as part of your team’s submission.